View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Dagstuhl Research Online Publication Server

Scalable Design Space Exploration via Answer Set Programming*

Philipp Wanko

University of Potsdam, Institute for Computer Science, Potsdam, Germany
wanko@cs.uni-potsdam.de

Abstract

The design of embedded systems is becoming continuously more complex such that the application of efficient high level design methods are crucial for competitive results regarding design time and performance. Recently, advances in Boolean constraint solvers for Answer Set Programming (ASP) allow for easy integration of background theories and more control over the solving process. The goal of this research is to leverage those advances for system level design space exploration while using specialized techniques from electronic design automation that drive new application-originated ideas for multi-objective combinatorial optimization.

1998 ACM Subject Classification D.1.6 Logic Programming

Keywords and phrases Answer Set Programming, System Synthesis, Multi-Objective Optimization

Digital Object Identifier 10.4230/OASIcs.ICLP.2016.23

1 Introduction and problem description

Embedded computing systems are application-specific computers. They typically have to satisfy among others real-time, power, and area constraints while being at the same time reliable and cost efficient. These often conflicting design goals can only be met because each embedded computing system is designed for a specific and thus restricted set of applications. Examples of embedded computing systems can be found in smart phones, automation systems, automotive electronics, medical devices, industrial automation systems, train control systems, etc. However, increasing application complexity paired with increasingly complex computing platforms hamper good design decisions and, thus, the optimization of the final product. As a consequence, new tools and methodologies are required, which permit to automatically and effectively explore design options at system level. The goal of this research is to leverage advances in Boolean constraint technology for system level design space exploration. In turn, specialized techniques from electronic design automation drive new application-originated ideas into multi-objective combinatorial optimization.

2 Background and overview of the existing literature

2.1 Design Space Exploration

(DSE) can be performed at various abstraction levels. The goal is always to identify an optimal implementation for the given set of applications. Depending on the level of abstraction, the considered applications can be as complex as a video decoder or as simple as a single logic

* This work was partially supported by DFG-SCHA-550/11.


© Philipp Wanko;
licensed under Creative Commons License CC-BY

operation. All possible implementations, also called design points, of the given applications define the *design space*, denoted by X. The problem of DSE is twofold [17]: (1) How to *evaluate* the quality of a single design point and (2) how to *cover* the design space during exploration? Again, depending on the abstraction level, the implementation can be as complex as a heterogeneous many-core system or as simple as a single logic gate. Our research targets the electronic system level and assumes a top-down design methodology [14]. At this level, applications are typically of the complexity of video decoders and computing platforms are many-core systems.

Starting from a given set of applications, a computing platform has to be *allocated* and the applications have to be *bound* and *scheduled* onto the allocated hardware resources [42]. Each application is typically assumed to consist of communicating tasks. During allocation, processing and interconnection resources including memories are selected and configured. In general, the result is a heterogeneous many-core computing platform, consisting of software-programmable processors and hardware accelerators interconnected by a network equipped with a memory hierarchy. During binding, tasks are assigned to processors and hardware accelerators as well as variables are assigned to memory locations. Moreover, transactions are routed on the interconnection resources. This step is crucial as *infeasible* implementations can be generated by binding two communicating tasks to virtually unconnected resources. The set of *feasible implementations* is denoted by $X_f \subseteq X$. Finally, scheduling resolves resource conflicts by precomputing either dedicated computing and communication times or priorities. According to the design decisions during allocation, binding, and scheduling, the set of applications can be refined. With respect to the resulting system decomposition, the design process can be continued at a lower level of abstraction [42]. The motivation for making as many decisions as possible during design time stems from the fact that embedded computing systems often have to guarantee many properties like safety, reliability, performance, etc. and, hence, demand for a high degree of predictability.

Depending on the made design decisions, the resulting system level implementations show different qualities. Typically, more than a single property is assessed to measure the quality of an implementation. Important properties are power and area consumption, throughput and response time, or mean time to failure. As an example, a video decoder implemented in a handheld system has to meet timing constraints in order to provide some quality of service, while simultaneously being power efficient. During DSE, appropriate evaluation methods have to be applied in order to *estimate* the quality (see below). Given a feasible implementation $x \in X_f$, its quality can be represented by a vector, which is commonly referred to as *quality vector* $f(x)$, where $f(x) = (f_1(x), \ldots, f_k(x))$ is a k-dimensional function consisting of k objective functions. Note that the notion of quality vectors, even if possible to compute, does not have any meaning for infeasible implementations.

Often several conflicting design goals are considered simultaneously. As a consequence, a set of *Pareto-optimal* solutions has to be found [36]. A solution is said to be Pareto-optimal, if it is not *dominated* by any other solution. For minimization problems and any two feasible implementations $x_1, x_2 \in X_f$, we say (cf. [45]):

$$\begin{array}{rrll} x_1 \succ x_2 & (x_1 \text{ dominates } x_2) & \text{iff} & \forall i : f_i(x_1) \leq f_i(x_2) \wedge \exists i : f_i(x_1) < f_i(x_2) \\ x_1 \sim x_2 & (x_1 \text{ is indifferent to } x_2) & \text{iff} & \forall i : f_i(x_1) = f_i(x_2) \\ x_1 \parallel x_2 & (x_1 \text{ is incomparable to } x_2) & \text{iff} & \exists i, j : f_i(x_1) > f_i(x_2) \wedge f_j(x_1) < f_j(x_2). \end{array}$$

This is illustrated in Figure 1(a): For design point x, the regions containing other design points by which x is dominated, which are dominated by x, and which are incomparable to x are shown. Typically, additional quality constraints $g_i(x) \leq b_i$ are imposed on each

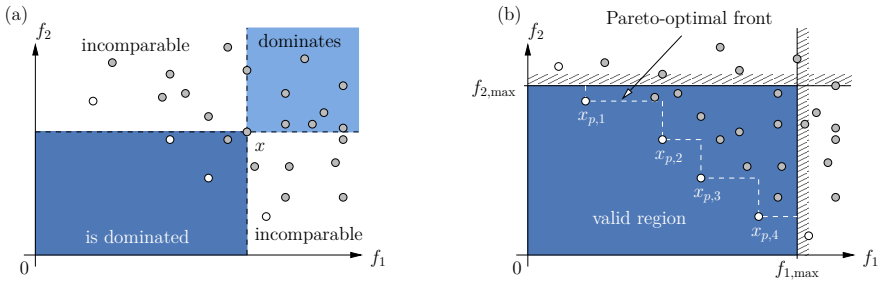

Figure 1 (a) Dominance in MOPs. (b) 2-dimensional objective space of a minimization problem.

implementation. In Figure 1(b) maximum values for both objectives f_1 and f_2 are given. Feasible implementations in X_f obeying all m constraints $g_1(x) \leq b_1, \ldots, g_m(x) \leq b_m$ are said to be *valid*, i.e., $X_v \subseteq X_f$. Among all valid implementations in X_v, the non-dominated ones are called *Pareto-optimal*. The set of Pareto-optimal implementations is denoted by $X_p \subseteq X_v$. The Pareto-optimal front Y_p is given by their corresponding quality vectors , i.e., $Y_p = \{f(x) \mid x \in X_p\}$. Without loss of generality, only minimization problems are considered in the proposal at hand.

2.2 Covering the Design Space

Nearly all approaches to DSE at the electronic system level follow the commonly known *Y-chart approach* [24] to represent the design space. Prominent examples are DOL [43], Daedalus [35], openDF [7], Octopus [5] and our approach SystemCoDesigner [23]. The Y-chart methodology starts by defining the *set of applications* and a *target architecture template*. Next, all possible *mappings* of tasks to resources in the target architecture template are defined. Such a *specification* is often represented by a graph structure [9]. Graph elements are annotated with implementation characteristics like task (worst case) execution times, required area, power, etc. These values are used by the objective functions and constraints to determine implementation properties. The values have to be provided in a preceding characterization phase [19, 25]. With this information, the exploration can be performed automatically by selecting resources from the target architecture template and by selecting the actual binding of tasks [9]. As state-of-the-art computing platforms are often heterogeneous many-core architectures including a Network-on-Chip (NoC) communication infrastructure, transaction routing is an additional complex synthesis task. A simple example is shown in Figure 2. The specification is shown in Figure 2(a). It consists of a task graph with three tasks (yellow circles) and a 3×3 meshed NoC architecture template. The mapping options are shown as green dashed edges m_1 to m_9. Figure 2(b) shows one feasible implementation. All resources and their interconnects are allocated. Task t_1 is bound onto resource $r_{1,1}$, t_2 onto $r_{2,3}$, and t_3 onto $r_{3,1}$. The transaction routing is shown by blue arrows.

The main benefit of the Y-chart approach is the opportunity to formulate the synthesis step as a selection problem [20]. As a consequence, DSE can be formally specified as a multi-objective combinatorial optimization problem [32], or for short Multi-objective Optimization Problem (MOP). With this abstraction, different optimization strategies such as *enumerative* optimization (e.g., exhaustive search), *deterministic* optimization, (e.g., hill climbing or branch and bound), or *stochastic* optimization, (e.g., simulated annealing, tabu search, or evolutionary algorithms) can be used to perform the search and, thus, to cover the design space.

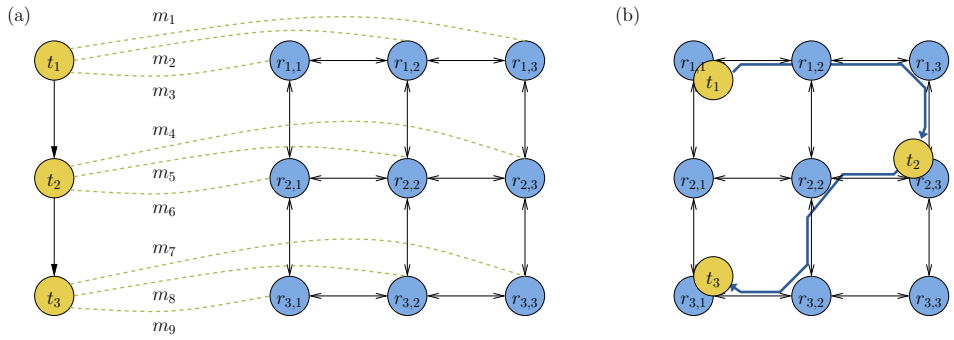

Figure 2 (a) Example of a specification and (b) a resulting implementation.

Due to the sheer size of typical design spaces, enumerative approaches are prohibitive. On the other hand, deterministic approaches often fail in the presence of non-linear objective functions and constraints. Among the stochastic approaches, population-based optimization strategies often perform best in the presence of MOPs [11]. The reason lies in the simultaneous approximation of the entire Pareto front by the individuals in the population, which preserves diversity among solutions while simultaneously converging to the true Pareto front by constantly improving good solutions. Examples of population-based optimization strategies are particle swarm optimization, ant colony optimization, and evolutionary algorithms. In particular, Multi-Objective Evolutionary Algorithms (MOEAs) [11] have been successful in the DSE domain [40]. Especially *elitist MOEAs*, which store the best found solutions in an external archive, show good properties in converging to the Pareto front [27].

Beside these advantages, MOEAs suffer from similar problems as many other stochastic optimization strategies: in the presence of design spaces only containing few feasible solutions, MOEAs spend most of their computing time in finding feasible solutions instead of optimizing feasible ones [40]. As a consequence, exploration time is not used efficiently. Due to the combinatorial nature of the optimization problem, this drawback can be alleviated by incorporating symbolic methods into MOEAs. In that case, symbolic methods are applied to perform the actual synthesis step, i.e., allocation, binding, and scheduling, which guarantees that feasible implementations are found if they exist. As a consequence, the MOEA now can focus on the optimization of feasible implementations. The idea of using symbolic methods in system synthesis is not new. In [34], a symbolic hardware/software partitioning based on Integer Linear Programming (ILP) is proposed. A symbolic system synthesis approach based on Binary Decision Diagrams (BDD) is presented in [33]. [20] presents an encoding as Boolean satisfiability problem (SAT), which enables system synthesis by programs known as SAT solvers. For this purpose, all design decisions (allocation, binding, routing, scheduling) are represented by Boolean variables z_i. Linear (feasibility) constraints $h_1(z_1, \ldots, z_l) \leq c_1, \ldots, h_n(z_1, \ldots, z_l) \leq c_n$ ensure that satisfying assignments represent design decisions leading to feasible implementations $x = \psi(z_1, \ldots, z_l) \in X_f$, where ψ is the decoding function that transforms a satisfying variable assignment into the corresponding feasible implementation. Nevertheless, all these approaches do not perform a Multi-Objective Optimization (MOO) as required by an unbiased DSE. An effective way to split the work among a SAT solver and the MOEA was presented in [31]: The MOEA triggers the SAT solver to generate a new satisfying assignment to the variables z_i (which represents a feasible implementation x) if possible. Afterwards, the MOEA computes the quality vectors and

checks the quality constraints $g_i(x) \leq b_i$. In all, the DSE can be modeled as an MOP:

$$\begin{array}{rll} \min & (f_1(\psi(z_1,\ldots,z_l)),\ldots,f_k(\psi(z_1,\ldots,z_l))) & \text{Checked by the MOEA} \\ \text{subject to} & g_1(\psi(z_1,\ldots,z_l)) \leq b_1,\ldots,g_m(\psi(z_1,\ldots,z_l)) \leq b_m & \text{Checked by the MOEA} \\ & h_1(z_1,\ldots,z_l) \leq c_1,\ldots,h_n(z_1,\ldots,z_l) \leq c_n & \text{Checked by the SAT solver} \end{array}$$

In the following, we will use the terms *quality constraints* and *feasibility constraints* in order to distinguish both kinds of constraints $g_i(\psi(z_1,\ldots,z_l)) \leq b_i$ and $h_i(z_1,\ldots,z_l) \leq c_i$, respectively.

2.3 Decision Procedures

As system design problems are becoming more and more stringent, the identification of feasible implementations in X_f is gaining importance in DSE. The formulation of the underlying system synthesis task as a Boolean satisfiability problem increases the interest in techniques for finding satisfying variable assignments. SAT solvers have been successfully applied to system verification in the past. Their success is largely boosted by the significant progress in Boolean constraint technology, often performing successfully even on huge instances with millions of variables and clauses. Though rooted in the classical DPLL algorithm, modern SAT solvers are mostly based on *Conflict-Driven Constraint Learning* (CDCL); see [8]. While both rely on unit propagation, CDCL basically extends DPLL by backjumping and constraint learning. Further essential supporting roles are played by dynamic conflict driven heuristics, lazy data structures, and restart policies (cf. [8]). Meanwhile, this advanced Boolean constraint technology is also used in many neighboring areas, like Maximum SAT (MAXSAT; [28]), Pseudo Boolean solving (PB; [38]), as well as Answer Set Programming (ASP; [4]).

A major weakness of the SAT-based approaches is, however, that reachability cannot be natively expressed. As a consequence, multi-hop communication has to be encoded as a sequence of communication steps, leading to unnecessarily huge (Pseudo-)Boolean formulas $h_i(z_1,\ldots,z_l) \leq c_i$ and, thus, long solving times. This is especially true for computing platforms with many different routing options, as we have shown for meshed-based NoCs [2, 3]. An approach similar to that of SAT yet directly supporting reachability is *Answer Set Programming*. ASP is an alternative approach to Boolean constraint solving tailored to knowledge representation and reasoning. As such, it combines a rich yet simple modeling language with advanced Boolean constraint technology. ASP's first-order language does not only offer scalability in terms of modeling and maintenance but moreover provides advanced language constructs like cardinality and weight constraints as well as optimization constructs. As a consequence, ASP's solving capacities do not only match the high performance of modern SAT solvers, but go well beyond clause-oriented satisfiability testing in integrating pseudo-Boolean constraints as well as optimization. The aforementioned representational edge of ASP over SAT is due a more stringent semantics that allows for more succinct Boolean problem representations [29]. Moreover, full-fledged ASP allows for solving all search problems in NP^{NP} in a uniform way. Given this expressiveness, it cannot only be used for computing feasible implementations in X_f but principally to even identify Pareto-optimal ones.

Finally, ASP solvers feature a whole spectrum of combinable reasoning modes surpassing satisfiability testing, among them, different forms of enumeration of solutions, intersection or union, as well as multi-criteria and -objective optimization. Notably, ASP supports polynomial space enumeration algorithms [12], which allows us to enumerate Pareto frontiers without risking an exponential blow-up in memory.

2.4 Evaluating Design Points

Independent of the selected optimization strategy, different design points can be constructed from the specification. Each of these solutions can be evaluated regarding feasibility and different objective functions. Important objective functions are power and area consumption, throughput and response time, or mean time to failure. In particular, assessing performance in terms of throughput and response time is often critical, as it is a foundation for determining other system properties like power efficiency and reliability. In our research, we are not going to develop new performance estimation methods. Instead, we rely on existing ones[1] and focus on a different problem: After evaluating a single design point x, it is known whether it obeys the quality constraints $g_i(x) \leq b_i$. If so, the implementation is called *valid*, otherwise *invalid*. All the above presented DSE approaches suffer from poor solving times if only a small fraction of feasible solutions is valid, i.e., $|X_v| << |X_f|$. The reason lies in an insufficiently strong feedback to the stochastic optimization method. Often a weak feedback exists by *punishing* invalid solutions by assigning uncompetitive objective values to them. However, as a consequence, invalid solutions might still be revisited again and again. A better strategy is to incorporate knowledge about the validity into the search process. Ideally, the decision procedure is used for this purpose. For constraints, which could be represented as continuous programming models, the classical Benders' decomposition [6] can be used. Benders' decomposition is a common method for solving mixed logical linear problems, where Boolean *indicator variables* are used to link different constrained problems. Thus, huge propositional logic formulas can be avoided. However, embedded systems design usually relies on combinatorial optimization. In this case Logic-Based Benders Decomposition (LBBD) could be used instead [21]. Its application to system synthesis is shown first in [39].

As LBBD only allows to test complete and consistent assignments of indicator variables, inconsistencies in linked programming models are thus lazily detected. This is avoided by using Satisfiability Modulo Theories (SMT; [8]) solvers, which permit working on partial assignments of indicator variables. Hence, larger regions of inconsistent assignments can be pruned and the search process is accelerated. This, however, requires monotonic constraints [1]. Unfortunately, this is not the case in embedded systems design, e.g., adding tasks to a partial implementation might decrease the response time. SMT is widely accepted in the domain of hardware and software verification. In SMT solving, a formula is tested for satisfiability with respect to a given *background theory*, e.g., Linear Real Arithmetics [41], Equality and Uninterpreted Functions [10]. SMT solvers are today typically *indirect* solvers, i.e., they are traditionally combinations of SAT solvers with background theory solvers. The SAT solver controls the solving process and assigns values to regular Boolean as well as indicator variables in the background theory. The background solver afterwards tries to find a corresponding variable assignment in the background theory to match the assignment of indicator variables. If a conflict is detected in the background theory, the reason could be learned by the SAT solver via the indicator variables.

In [30], the usage of SMT solving in systems synthesis is shown. The authors use a latency computation as background theory and perform optimization by a branch and bound strategy incorporated into the SMT solver. The system model, however, is based on a simple application model and communication architecture. Moreover, the proposed optimization is only applicable to single-objective optimization problems. Another SMT-based approach

[1] To be more precise, we consider Scenario-Aware Data Flow Graphs (SA-DFGs) [13] as application model of computation. For SA-SDFGs a performance analysis based on (max,+)-algebra exists [15, 16].

to synthesis is proposed in [22]. The underlying platform is a time-triggered architecture. As background theory, the authors use linear arithmetics for adding worst-case execution times. Thus, they stick with linear (monotonic) quality constraints. It was shown in [37] how to use Modular Performance Analysis (MPA) [44] as background theory to test real-time constraints, and, hence, how to integrate non-monotonic quality constraint checking into a SAT-based symbolic synthesis approach. Another group shows in [26] how SMT-solving with MPA as background theory can be used to compute processor frequency settings to meet delay, buffer, and energy constraints.

2.5 Assessing Exploration Quality

When developing different optimization approaches, it becomes mandatory to define appropriate performance measures to compare these approaches. In MOO, there are two different goals which must be considered when assessing optimization strategies: (1) The *convergence* towards the true Pareto-optimal front and (2) the *diversity* of the found non-dominated solutions [11]. In [45], a framework for comparing different performance assessment methods for multi-objective optimizers is presented. As a key result, it has been shown that binary quality indicators have to be used in order to decide whether an approximation set computed by an optimization strategy is better than one computed by another optimization strategy.

One of the best known binary quality indicators is ϵ-dominance [27]: A quality vector a is said to *weakly ϵ-dominate* (in a minimization problem) a quality vector b, denoted by $a \succeq_\epsilon b$, if and only if $a \succeq \epsilon \cdot b$. By scaling quality vector b by a factor ϵ, quality vector a is superior to quality vector b. Complementary to ϵ-dominance, which is primarily used to measure convergence, we use *entropy* [18] to measure diversity and keep diversity high when selecting representative design points.

3 Goal of the research

The state-of-the-art section has presented in detail that today's Design Space Exploration (DSE) approaches at the Electronic System Level (ESL) have the following drawbacks:

1. Often complex multi-hop communication is not supported. This neglects state-of-the-art computing platforms like many-core systems. Approaches that support multi-hop communication fail in the presence of computing platforms with many routing options, as can be typically found in Networks on Chip (NoCs).
2. Typically, no strong feedback from constraint checking to the optimization strategy exists. As a consequence, invalid solutions might be revisited again and again. This significantly lowers the exploration performance, which is particularly problematic when designs are becoming more stringent.
3. The specification of the target architecture template and all mapping options is a time consuming task. On the other hand, this specification allows to formulate the system synthesis problem as a selection problem and, thus, the automatic DSE.

From these shortcomings, we derive the following objectives from the perspective of electronic design automation:

O1: *Accelerate* DSE by integrating routing computation and dominance checking into the decision procedure.

O2: *Extend the applicability* of DSE at ESL by tightly incorporating non-monotonic quality constraint checking.

O3: *Improve the usability* of DSE at ESL by moving from selective methods to novel generative approaches.

From the viewpoint of Answer Set Programming (ASP), the general objective is to *invent new solving strategies* inspired from novel application-specific problems. More specifically, (i) *the integration* of application-specific knowledge and strategies into ASP solving should be *improved* and (ii) *the applicability* of ASP towards robust Multi-Objective Optimization (MOO) should be *extended*.

4 Current status and preliminary results of the research

Right now, the main focus is on exploring technologies and techniques to efficiently implement *O1-O3*. While no new publications have been made for Design Space Exploration specifically, the following contributions laid the groundwork for future applications:

Theory Solving made easy with Clingo 5 by M. Gebser, R. Kaminski, B. Kaufmann, M. Ostrowski, T. Schaub, and P. Wanko to appear as Technical Communication in ICLP'16. The new theory framework in *Clingo 5* allows for a tight coupling of decision procedures and efficient Boolean constraint solving. As an example, Difference Logic is implemented in the paper which is an efficient theory to implement temporal constraints which can be used to encode the scheduling needed in DSE.

Computing Diverse Optimal Stable Models by J. Romero, T. Schaub, and P. Wanko to appear as Technical Communication in ICLP'16. The paper introduces a system to pose queries over and enumerate diverse optimal solutions. This can be used for covering the Design Space and finding representative Pareto optimal solutions.

5 Open issues and expected achievements

We expect to achieve the following during our research:

1. ASP-Based Synthesis that includes Encodings for Many-Core Synthesis of Streaming Applications
2. Application-Specific Search and Enumeration Methods Based on ASP
3. Application-Specific Multi-Objective Optimization based on ASP
4. Application-Specific Theory Solving

References

1 Santosh G. Abraham, B. Ramakrishna Rau, and Robert Schreiber. Fast Design Space Exploration Through Validity and Quality Filtering of Subsystem Designs. Technical report, Hewlett Packard, Compiler and Architecture Research, HP Laboratories Palo Alto, July 2000.

2 B. Andres, M. Gebser, M. Glaß, C. Haubelt, F. Reimann, and T. Schaub. A combined mapping and routing algorithm for 3D NoCs based on ASP. In C. Haubelt and D. Timmermann, editors, *Sechzehnter Workshop für Methoden und Beschreibungssprachen zur Modellierung und Verifikation von Schaltungen und Systemen (MBMV'13)*, pages 35–46. Institut für Angewandte Mikroelektronik und Datentechnik, Universität Rostock, 2013.

3 B. Andres, M. Gebser, M. Glaß, C. Haubelt, F. Reimann, and T. Schaub. Symbolic system synthesis using answer set programming. In P. Cabalar and T. Son, editors, *Proceedings of the Twelfth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'13)*, volume 8148 of *Lecture Notes in Artificial Intelligence*, pages 79–91. Springer, 2013.

4 C. Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press, 2003.

5 T. Basten, M. Hendriks, L. Somers, and N. Trcka. Model-Driven Design-Space Exploration for Software-Intensive Embedded Systems. In *Proceedings of the International Conference on Formal Modeling and Analysis of Timed Systmes (FORMATS)*, pages 1–6, 2012.

6 J. F. Benders. Partitioning Procedures for Solving Mixed-Variables Programming Problems. *Numerische Mathemathik*, 4(3):238–252, 1962.

7 S. Bhattacharyya, G. Brebner, J. Janneck, J. Eker, C. von Platen, M. Mattavelli, and M. Raulet. OpenDF: A Dataflow Toolset for Reconfigurable Hardware and Multicore Systems. *ACM SIGARCH Computer Architecture News*, 36(5):29–35, 2009.

8 A. Biere, M. Heule, H. van Maaren, and T. Walsh, editors. *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2009.

9 T. Blickle, J. Teich, and L. Thiele. System-Level Synthesis Using Evolutionary Algorithms. In *Design Automation for Embedded Systems*, 3, pages 23–62. 1998.

10 J. R. Burch and D. L. Dill. Automatic Verification of Pipelined Microprocessor Control. In *Proceedings of the International Conference on Computer Aided Verification (CAV)*, pages 68–80, 1994.

11 K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons, Inc., Chichester, New York, Weinheim, Brisbane, Singapore, Toronto, 2001.

12 M. Gebser, B. Kaufmann, A. Neumann, and T. Schaub. Conflict-driven answer set enumeration. In C. Baral, G. Brewka, and J. Schlipf, editors, *Proceedings of the Ninth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'07)*, volume 4483 of *Lecture Notes in Artificial Intelligence*, pages 136–148. Springer, 2007.

13 M. Geilen and S. Stuijk. Worst-Case Performance Analysis of Synchronous Dataflow Scenarios. In *Proceedings of the Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, pages 125–134, 2010.

14 A. Gerstlauer, C. Haubelt, A. D. Pimentel, T. P. Stefanov, D. D. Gajski, and J. Teich. Electronic System-Level Synthesis Methodologies. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 28(10):1517–1530, 2009.

15 A. H. Ghamarian, M. C. W. Geilen, S. Stuijk, T. Basten, A. J. M. Moonen, M. J. G. Bekooij, B. D. Theelen, and M. R. Mousavi. Throughput Analysis of Synchronous Data Flow Graphs. In *Proceedings of the International Conference on Application of Concurrency to System Design (ACSD)*, pages 25–36, 2006.

16 A. H. Ghamarian, S. Stuijk, T. Basten, M. C. W. Geilen, and B. D. Theelen. Latency Minimization for Synchronous Data Flow Graphs. In *Proceedings of the Euromicro Conference on Digital System Design Architectures, Methods and Tools (DSD)*, pages 189–196, 2007.

17 M. Gries. Methods for Evaluating and Covering the Design Space during Early Design Development. *Integration, The VLSI Journal*, 38(2):131–183, 2004.

18 S. Gunawan, Ali Farhang-Mehr, and Shapour Azarm. Multi-Level Multi-Objective Genetic Algorithm Using Entropy to Preserve Diversity. In *Proceedings of the International Conference on Evolutionary Multi-Criterion Optimization (EMO)*, pages 148–161, 2003.

19 W. Haid, M. Keller, K. Huang, I. Bacivarov, and L. Thiele. Generation and Calibration of Compositional Performance Analysis Models for Multi-Processor Systems. In *Proceedings of the International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation (SAMOS)*, pages 92–99, 2009.

20 C. Haubelt, J. Teich, R. Feldmann, and B. Monien. SAT-Based Techniques in System Design. In *Proceedings of the Design, Automation and Test in Europe (DATE)*, pages 1168–1169, 2003.

21 J. N. Hooker and G. Ottosson. Logic-Based Benders Decomposition. *Mathematical Programming*, 96(1):33–60, 2003.

22 E. Jackson, E. Kang, M. Dahlweid, D. Seifert, and T. Santen. Components, Platforms and Possibilities: Towards Generic Automation for MDA. In *Proceedings of the International Conference on Embedded Software (EMSOFT)*, pages 39–48, 2010.

23 J. Keinert, M. Streubühr, T. Schlichter, J. Falk, J. Gladigau, C. Haubelt, J. Teich, and M. Meredith. SystemCoDesigner – An Automatic ESL Synthesis Approach by Design Space Exploration and Behavioral Synthesis for Streaming Applications. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 14(1):1–23, 2009.

24 B. Kienhuis, E. Deprettere, K. Vissers, and P. van der Wolf. An Approach for Quantitative Analysis of Application-Specific Dataflow Architectures. In *Proceedings of the Conference on Application-Specific Systems, Architectures and Processors (ASAP)*, pages 338–349, 1997.

25 R. Kiesel, M. Streubühr, C. Haubelt, O. Löhlein, and J. Teich. Calibration and Validation of Software Performance Models for Pedestrian Detection Systems. In *Proceedings of the International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation (SAMOS)*, pages 182–189, 2011.

26 P. Kumar, D. B. Chokshi, and L. Thiele. A Satisfiability Approach to Speed Assignment for Distributed Real-Time Systems. In *Proceedings of the Design, Automation and Test in Europe (DATE)*, pages 749–754, 2013.

27 M. Laumanns, L. Thiele, K. Deb, and E. Zitzler. Combining Convergence and Diversity in Evolutionary Multi-Objective Optimization. *Evolutionary Computation*, 10(3):263–282, 2002.

28 C. Li and F. Manyà. MaxSAT. In Biere et al. [8], chapter 19, pages 613–631.

29 V. Lifschitz and A. Razborov. Why are there so many loop formulas? *ACM Transactions on Computational Logic*, 7(2):261–268, 2006.

30 W. Liu, Z. Gu, J. Xu, X. Wu, and Y. Ye. Satisfiability Modulo Graph Theory for Task Mapping and Scheduling on Multiprocessor Systems. *IEEE Transactions on Parallel and Distributed Systems*, 22(8):1382–1389, 2011.

31 M. Lukasiewycz, M. Glaß, C. Haubelt, and J. Teich. SAT-Decoding in Evolutionary Algorithms for Discrete Constrained Optimization Problems. In *Proceedings of the Congress on Evolutionary Computation*, pages 935–942, 2007.

32 M. Lukasiewycz, M. Glaß, C. Haubelt, and J. Teich. Efficient Symbolic Multi-Objective Design Space Exploration. In *Proceedings of the Asia and South Pacific Design Automation Conference (ASPDAC)*, pages 691–696, 2008.

33 S. Neema. *System Level Synthesis of Adaptive Computing Systems*. PhD thesis, Vanderbilt University, Nashville, Tennessee, 2001.

34 R. Niemann and P. Marwedel. An Algorithm for Hardware/Software Partitioning Using Mixed Integer Linear Programming. *Design Automation for Embedded Systems*, 2(2):165–193, 1997.

35 H. Nikolov, M. Thompson, T. Stefanov, A. D. Pimentel, S. Polstra, R. Bose, C. Zissulescu, and E. F. Deprettere. Daedalus: Toward Composable Multimedia MP-SoC Design. In *Proceedings of the Design Automation Conference (DAC)*, pages 574–579, 2008.

36 V. Pareto. *Cours d'Économie Politique*, volume 1. F. Rouge & Cie., 1896.

37 F. Reimann, M. Glaß, C. Haubelt, M. Eberl, and J. Teich. Improving Platform-Based System Synthesis by Satisfiability Modulo Theories Solving. In *Proceedings of the Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, pages 135–144, 2010.

38 O. Roussel and V. Manquinho. Pseudo-Boolean and cardinality constraints. In Biere et al. [8], chapter 22, pages 695–733.

39 N. Satish, K. Ravindran, and K. Keutzer. A Decomposition-Based Constraint Optimization Approach for Statically Scheduling Task Graphs with Communication Delays to Multiprocessors. In *Proceedings of the Design, Automation and Test in Europe (DATE)*, pages 57–62, 2007.

40 T. Schlichter, C. Haubelt, and J. Teich. Improving EA-based Design Space Exploration by Utilizing Symbolic Feasibility Tests. In *Proceedings of Genetic and Evolutionary Computation Conference> (GECCO)*, pages 1945–1952, 2005.

41 H. M. Sheini and K. A. Sakallah. A Scalable Method for Solving Satisfiability of Integer Linear Arithmetic Logic. In *Theory and Applications of Satisfiability Testing*, pages 241–256, 2005.

42 J. Teich and C. Haubelt. *Digitale Hardware/Software-Systeme – Synthese und Optimierung*. Springer, Berlin, Heidelberg, 2007. 2. erweiterte Auflage.

43 L. Thiele, I. Bacivarov, W. Haid, and K. Huang. Mapping Applications to Tiled Multiprocessor Embedded Systems. In *Proceedings of the International Conference on Application of Concurrency to System Design (ACSD)*, pages 29–40, 2007.

44 L. Thiele and E. Wandeler. Performance Analysis of Distributed Embedded Systems. In *Embedded Systems Handbook*, pages 15.1–15.18. CRC Press, Boca Raton, FL, 2006.

45 E. Zitzler, L. Thiele, M. Laumanns, C. Fonseca, and V. Grunert da Fonseca. Performance Assessment of Multiobjective Optimizers: An Analysis and Review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132, 2003.